MATTINGLY, STANGER & MALUR, P.C.

JOHN R. MATTINGLY*
DANIEL J. STANGER
SH[illegible] MALUR*

[illegible] STOCKMAN
[illegible]
[illegible] KETCHUM
[illegible] BRICKNER
[illegible] Patent Agents
[illegible]ship Other Than Virginia

ATTORNEYS AT LAW
104 EAST HUME AVENUE
ALEXANDRIA, VIRGINIA 22301

(703) 684-1120

PATENT, TRADEMARK
AND COPYRIGHT LAW

FACSIMILE (703) 684-1157

09/22/00
jc841 U.S. PTO
09/667716
09/22/00
09667716 .092200

Date: September 22, 1900

Attorney Docket No. H-944

To: Assistant Commissioner for Patents
Washington, D.C. 20231

Sir: Transmitted herewith for filing is the patent application of:

Inventor: SEE ATTACHED LIST (K. SHIMADA)

For:
MULTIPROCESSOR MACHINE AND CACHE CONTROL METHOD

Enclosed are:

- [X] 11 Sheets of Drawings
- [ ] This application is being filed without an executed Declaration.
- [ ] Priority is claimed from ______ Application No. ______ filed ______. [ ] A certified copy is attached herewith.
- [X] Copies of the disclosure documents listed on the attached PTO 1449 form and [X] discussed in the specification or [X] attached Information Disclosure Statement.
- [ ] A verified statement to establish small entity status under 37 CFR 1.9 and 1.27.
- [X] Specification: Abstract X, Description 21 pages; and 14 claim(s).
- [ ] Preliminary Amendment.
- [X] Executed Declaration.

The filing fee is calculated as shown below:

For:	No. Filed	No. Extra	Small Entity Rate	Small Entity Fee		Large Entity Rate	Large Entity Fee
Basic Fee				$ 345	OR		$ 690
Total Claims	14 -20 =	* 0	x 9	$		x 18	$ 0
Indep Claims	4 -3 =	* 1	x 39	$		x 78	$ 78
[ ] Multiple Dependent Claim (s)			+ 130	$		+ 260	$ 0
			Total	$		Total	$ 768

* If difference is less than zero then enter '0' in second column

- [X] A check in the amount of $ 768.00 is enclosed for the filing fee.
- [X] The Commissioner is hereby authorized to charge any additional fees that may be required to Deposit Account No. 50-1417.

Respectfully Submitted,

By: John R. Mattingly
Registration No. 30,293

H-944
310000136US1

LIST OF INVENTOR'S NAME AND ADDRESS

Kentaro SHIMADA, Tokyo, JAPAN.

H-944
310000136US1

Title of the Invention

MULTIPROCESSOR MACHINE AND CACHE CONTROL METHOD

Inventor

Kentaro SHIMADA

Title of the invention

Multiprocessor machine and cache control method

Background of the invention

The present invention relates to a technology for controlling cache in multiprocessor machines. More specifically, the present invention relates to cache controller in chip multiprocessors.

An example of a conventional technology for controlling multiprocessor cache is a technology that seeks to increase speed by reducing control hardware and control signal traffic from control data used to maintain consistency in data shared between the plurality of processors. Examples of this technology are described in Japanese laid-open patent publication number Hei 11-272557, Japanese laid-open patent publication number Hei 09-293060, and Japanese laid-open patent publication number Hei 08-263374.

With LSI chips, the data transfer between the chip and external components is restricted by the physical limitation of the number of chip pins. Thus, it would be desirable to reduce the communication between the chip and external components as much as possible. Thus, with chip multiprocessors in which two or more processors and a cache are integrated on an LSI chip, cache control must be performed to

reduce the communication between the on-chip cache and external components.

In the conventional technology described above, the communication between the chip and external components cannot be reduced. On the other hand, the object of the conventional technology to simplify and increase the speed of control performed to maintain cache consistency is not a major issue since a large amount of data can be communicated between the on-chip processors.

Summary of the invention

In multiprocessor machines and chip multiprocessor systems in particular, the object of the present invention is to reduce data communication between the LSI chip and external components and to avoid restrictions in communication volume resulting from the LSI pin count. The overall system performance can be improved by achieving these objects.

In order to achieve these objects, a multiprocessor machine according to the present invention includes a plurality of processors and a first cache shared by said plurality of processors. The first cache is controlled so that, when storing data, it gives priority to data accessed by at least two processors of the plurality of processors.

Also, second caches are used by each of the plurality of processors. If data stored in the second cache is accessed by a processor other than the processor owning the second cache, priority is not given to the second cache when storing data.

Also, the plurality of processors and the first cache are integrated on a single LSI chip. Also, the plurality of processors, the first cache, and the second cache are integrated on a single LSI chip.

Furthermore, first selecting means gives priority to areas containing data not accessed by at least two processors of the plurality of processors when selecting an area in the first cache to store new data.

Furthermore, second selecting means gives priority to areas containing data accessed by a processor other than the processor owning the second cache when selecting an area in the second cache to store new data.

Also, in order to achieve the objects described above, a method for controlling cache according to the present invention includes: a first step evaluating whether data stored in a cache shared by a plurality of processors is accessed by at least two processors from the plurality of processors; a second step selecting an area determined by the first step to not be accessed by at least two processors when storing new data to the cache; a third step selecting an area in the first cache if no

area can be selected in the second step; and a fourth step storing the new data in an area of the first cache selected by either the second step or the third step.

Also, in the third step, an area in the first cache containing data with the lowest number of accessing processors of the plurality of processors is selected.

Also, the present invention includes: a first step evaluating whether data stored in a second cache associated with one of a plurality of processors was accessed by a processor other than a processor associated with the second cache; a second step selecting an area containing data determined in the first step to have been accessed by another processor when new data is stored in the second cache; a third step selecting an area of the second cache if no area can be selected in the second step; and a fourth step storing new data in an area of the second cache selected by either the second step or the third step.

Brief description of the drawings

Fig. 1 is a example of drawing of the system architecture of a multiprocessor machine according to the present invention.

Fig. 2 is a example of flowchart for the purpose of describing a method for controlling cache according to the present invention.

Fig. 3 is a drawing showing the system architecture of a multiprocessor machine according to a comparative example of the present invention.

Fig. 4 is a flowchart of a method for controlling cache according to a comparative example of the present invention.

Fig. 5 is a drawing for the purpose of describing the operations of a system according to the present invention.

Fig. 6 is a drawing for the purpose of describing the operations of a system according to a comparative example of the present invention.

Fig. 7 is a drawing of another system architecture of a multiprocessor system according to the present invention.

Fig. 8 is a flowchart of another method for controlling cache according to the present invention.

Fig. 9 is a flowchart of a method for controlling cache according to a comparative example of the present invention.

Fig. 10 is a drawing for the purpose of describing another system according to the present invention.

Fig. 11 is a drawing for the purpose of describing the operations of a system according to a comparative example of the present invention.

Detailed description of the preferred embodiments

The following is a description of the embodiments of the present invention.

Fig. 1 is a drawing showing the system architecture of a chip multiprocessor equipped with a shared cache. A chip multiprocessor 1 includes four processors 10a - 10d and a shared cache 4. In Fig. 1, the chip multiprocessor 1 is connected to a main storage 2, but it would also be possible to provide a separate cache interposed between these two elements.

The four processors 10a - 10d share the cache 4 using a common data bus 102 and a common address bus 101 on the chip of the chip multiprocessor 1. The shared cache 4 includes: two-way tag and data blocks 11a, 11b; an LRU memory 12; a way selector 13 to the common data bus 102; a sharing controller 14; tag address comparators 15a, 15b; a hit check controller 16; a replacement controller 17; and a way selector 18 to an external data bus. The tag and data blocks 11a, 11b are formed from a plurality of sets selected ~~from~~ based on part a ~~section~~ of the memory address. In addition to a tag address and data, each set stores a valid bit V, a shared bit S, a dirty bit D, and a processor number P for the processor that last accessed the data. ~~In the tag and data blocks 11a, 11b~~ Each set includes two groups of tag addresses, data, valid bits V,

shared bits S, dirty bits D, and processor numbers P in the tag and data blocks 11a, 11b, and each of these groups is known as a way. For each set, the LRU memory 12 ~~stores~~ records the way that was most recently accessed.

The four processors 10a - 10d access data using the common address bus 101 and the common data bus 102. When data is accessed, a ~~section~~ part of the data address in the common address bus 101 is used to select a set to be referenced. Tag addresses corresponding to the selected set is output from the tag and data blocks 11a, 11b, and the tag address comparators 15a, 15b compare the remaining sections of the data address on the address bus 101. At the same time, the valid bit V values are read from the tag and data blocks 11a, 11b. If there is a match from either the tag address comparator 15a or the tag address comparator 15b, and if the value of the valid bit V for the corresponding tag and data block 11a or tag and data block 11b is 1, the hit check controller 16 determines that there is a hit.

When the hit check controller 16 determines that there is hit, the way selector 13 to the common bus is controlled and, in the case of a data read operation, the corresponding data from the tag and data block 11a or the tag and data block 11b is output to the common data bus 102. In the case of a data write operation, the data on the common data bus 102 is written to the corresponding data section in the tag

and data block 11a or the tag and data block 11b. At the same time, the dirty bit D is also set. When either reading or writing, the contents of the LRU memory 12 are changed to indicate the way for which a hit was determined. Also, the sharing controller 14 compares the processor number of the accessing processor 10a - 10d with the corresponding processor number P in the tag and data block 11a or the tag and data block 11b. If they are different, the corresponding shared bit S of the tag and data block 11a or the tag and data block 11b is set. The processor number of the accessing processor 10a - 10d is notified to the sharing controller 14 using a section of the address bus 101. Then, when the operation for the shared bit S is completed, the processor number is written to the corresponding processor number P in the tag and data block 11a or the tag and data block 11b.

If the hit check controller 16 determines a miss, the result is sent to the replacement controller 17. The replacement controller 17 reads the corresponding data from the main storage 2 and saves it to the tag and data block 11a or the tag and data block 11b in the set selected using a ~~section~~ part of the address on the address bus 101. When doing this, if the valid bit V for either the tag and data block 11a or the tag and data block 11b is 0, then the corresponding block is selected. In other words, new data is stored where invalid data was stored. If the valid bits V for both

K.S.

are 1, the shared bits S are examined. If either bit is 0, the corresponding block is selected. In other words, the new data is stored where data not shared between processors was stored.

With the operations described above, data shared between at least two processors is not removed from the cache, thus allowing the data transfer with the main storage 2 to be reduced. If the shared bits S are both 1 or the shared bits S are both 0, the contents of the LRU memory 12 for the corresponding set is examined and a block is chosen so that the side having the earlier access is removed from the cache. The replacement controller 17 controls the way selector 18 so that the selected block from the tag and data blocks 11a, 11b is connected to the main storage 2. The dirty bit D of the selected block from the tag and data blocks 11a, 11b is checked, and if the dirty bit D is set the current contents of the corresponding data section is written to the main storage 2. Next, for addresses determined by the hit check controller 16 to be a miss, the corresponding data is read from the main storage 2 and is written to the selected block from the tag and data blocks 11a, 11b. Finally, the valid bit V is set, the dirty bit D and the shared bit S are reset, and the corresponding processor number is written to the processor number P. Then, operations similar to those performed for hits are performed and the reading or

writing of the data is completed.

Fig. 2 is a flowchart showing an example of a method for controlling shared cache according to the present invention. The number of ways in the example shown in Fig. 2 is also two. In Fig. 2, when access to the shared cache begins, step 201 checks to see if there is a hit or not. If there is a hit at step 202, control goes to step 212. If there is a miss, control goes to step 203, where the ways are checked to see if there is a way with the valid bit V set to 0. If there is a way with V set to 0, the way is selected at step 205. If there is no way with V set to 0, step 204 checks to see if, out of the two ways, one has the shared bit S set to 0. If S=0, control goes to step 206 and the way with S=0 is selected. If both ways have S=0 or S=1, the LRU is used to select the way that was used earliest. If three or more ways are to be used, the LRU can be used to select the way that was used earliest if there are at least two ways with S=0.

At step 208, the selected way is checked to see if the dirty bit D is set to 1 or not. If it is set to 1, control goes to step 209, and the contents of the way are written outside the chip, to the corresponding address in the main storage. Control then goes to step 210. If the dirty bit D is set to 0, control goes directly to step 210. At step 210, the newly accessed address contents from the main storage are read and stored to the selected way. Next, at step 211, the valid bit V is set to 1, the shared bit S is set to

0, the dirty bit D is set to 0, and the processor number P is set to the processor number of the processor performing the access. Control then proceeds to step 212 as in the case of a cache hit.

Step 212 checks to see if the access is a read or a write. If the access is a read operation, control goes to step 213. The corresponding data is read from the shared cache and is output to the common data bus. If the access is a write operation, control goes to step 214, where the write data output from the processor is written to the shared cache from the common data bus. Then, at step 215, the dirty bit D is set to 1.

Step 216 is reached from step 213 or step 215, and the processor number of the accessing processor is compared with the recorded processor number P. If the values are different, the shared bit S is set to 1 at step 217. Finally, at step 218, the processor number of the accessing processor is stored in the processor number P and the shared cache accessing operations are completed.

The following is a detailed description of the operations performed by the present invention compared to other systems.

Fig. 3 is a sample system (comparative example) prepared for comparison with an example of a multiprocessor machine according to the present invention

shown in Fig. 1. Compared to the present invention shown in Fig. 1, tag and data blocks 31a, 31b in a shared cache 6 do not contain shared bits S or processor numbers P, and there is also no sharing controller 14. The operations are similar to corresponding operations performed in Fig. 1 except that there are no operations relating to the shared bits S, the processor numbers P, or the sharing controller 14.

Fig. 4 is a flowchart showing the shared cache control method prepared for comparison with the shared cache control method according to the present invention as shown in Fig. 2. Compared to the example of the present invention indicated in Fig. 2, steps corresponding to steps 204, 206 for selecting ways using the shared bits S are absent. Also, step 411 from Fig. 4 does not include the operations relating to shared bits S as in step 211 from Fig. 2. Furthermore, steps corresponding to steps 216, 217, 218 for setting the shared bit S using the processor number P and storing the accessing processor number are absent.

With these changes, the present invention operates as follows and provides the desired advantages.

Fig. 5 and Fig. 6 are figures for describing the operations of the present invention shown in Fig. 1 and the operations of the comparative example shown in Fig. 3. Fig. 5 shows an example of an operation performed by the present invention.

Processors a, b access shared area addresses 0100 - 0107, and processors c, d access private areas 2100 - 2107, 3100 - 3107, 4100 - 4107, and 5100 - 5107 in the sequence shown in the figure. To simplify the description, these addresses will correspond to a single set in the shared cache 4. Fig. 5 shows, for each point in time, the addresses of the main storage 2 cached in the set in the tag and data blocks 11a, 11b. In the example of the present invention shown in Fig. 5, the total size of the data transferred from the main storage 2 in the operation shown in the figure is 40 bytes. In contrast, Fig. 6 shows the operations performed by the comparative example from Fig. 3 for the same data accesses as the example shown in Fig. 5. In Fig. 6, the total size of the data transferred from the main storage 2 is 56 bytes, which is 1.4 times the size from Fig. 5.

Fig. 7 is another example of a multiprocessor machine according to the present invention in which private caches 7a - 7d are added to the processors 10a - 10d. In Fig. 7, the shared cache 4 is similar to the one from the architecture shown in Fig. 1. The private caches 7a - 7d are formed identically and include: two-way tag and data blocks 71a, 71b; an LRU memory 72; a processor way selector 73; a snooping/sharing controller 74; tag address comparators 75a, 75b; a hit check controller 76; a replacement controller 77; and a way selector 78 for the shared cache

and external connections. In addition to tag addresses and data, the tag and data blocks 71a, 71b store valid bits V, shared bits S, and dirty bits D. The LRU memory 72 store the most recently accessed way in each set.

The following is a description of operations performed using the private cache 7a added to the processor 10a as an example. When the processor 10a accesses data, a ~~section~~ part of the accessed data address is used to select a set to be referenced. K.S. Tag addresses are output from the tag and data blocks 71a, 71b for the selected set, and the tag address comparators 75a, 75b compares these with the remaining section of the data address. At the same time, the valid bits V are read from the tag and data blocks 71a, 71b. If the tag address comparator 75a or the tag address comparator 75b show a match and the valid bit V from the corresponding tag and data block 71a or tag and data block 71b is 1, then the hit check controller 76 determines that there is a hit. When the hit check controller 76 determines that there is a hit, the processor way selector 73 is controlled and the corresponding data in the tag and data block 71a or the tag and data block 71b is output if the operation is a data read operation. If the operation is a data write operation, the shared bit S is checked. If it is set to 1, the snooping/sharing controller 74 is notified. The snooping/sharing controller 74 receives the notification and outputs the data write

address to the common address bus, and a request is made to invalidate the corresponding data in the private caches 7b - 7d of the other processors. Then, the hit check controller 76 resets the shared bit S, writes the corresponding data to the tag and data block 71a or the tag and data block 71b, and sets the dirty bit D. For both read and write operations, the contents of the LRU memory 72 are updated to indicate the way that was determined to be a hit.

If the hit check controller 76 determines that the access is a miss, the result is notified to the replacement controller 77. The replacement controller 77 refers to the corresponding data in the shared cache 4 or the main storage 2. If the corresponding data is stored in the shared cache 4 and the shared bit S in the shared cache 4 is set, then the data in the shared cache 4 is referenced and the contents of the tag and data block 71a or the tag and data block 71b are not updated. Otherwise, the corresponding data is read from the shared cache or the main storage 2 and stored using the operations described below ~~via the way to~~ either the tag and data block 71a or the tag and data block 71b into the set selected based on ~~using~~ the part ~~section~~ of the K, S, data address, in.

First, if either valid bit V from the corresponding set in the tag and address block 71a or the tag and address block 71b is 0, that block is selected. If both valid

bits V are set to 1, the shared bit S is checked and, if either is set to 1, that block is selected. This allows data in the shared cache 4 that is not shared to be kept while allowing effective use of the fixed data capacity in the tag and data blocks 71a, 71b. If both shared bits S are set to 0, the contents of the LRU memory 72 corresponding to the set are checked and the one with the older access time is selected.

The replacement controller 77 controls the way selector 78 so that the selected block from the tag and data block 71a or the tag and data block 71b is connected to the shared cache 4 or the main storage 2. Also, the dirty bit D of the selected block from the tag and data block 71a or the tag and data block 71b is checked, and if the dirty bit D is set the current contents of the data section is written back to the shared cache 4 or the main storage 2. Next, for addresses determined to be misses by the hit check controller 76, the corresponding data is read from the shared cache 4 or the main storage 2 and is written to the tag and data block 71a or the tag and data block 71b. Finally, the valid bit V is set and the dirty bit D and the shared bit S are reset. Then, operations similar to those performed when there is a hit are performed, and the reading or writing of data is completed.

The snooping/sharing controller 74 monitors, via the common address bus 101, accesses to the shared cache 4 and the main storage 2 from the private caches

71b - 71d of the other processors 10b - 10d. if an invalidation request is output from another private cache 71b - 71d, the corresponding address in the tag and data blocks 71a, 71b is checked, and if the data for the corresponding address is stored, the replacement controller 77 and the like are controlled to invalidate this data. Also, when another private cache 7b - 7d accesses the shared cache 4 or the main storage 2 via the common address bus 101, the corresponding address in the tag and data blocks 71a, 71b is checked, and if the corresponding the data for the corresponding address is stored, the corresponding shared bit S is set to 1. Also, if the dirty bit D for the corresponding data is set, this data is output to the common data bus 102 instead of the shared cache 4 or the main storage 2. Furthermore, this data is also written by the replacement controller 77 to the shared cache 4 or the main storage 2, and the dirty bit D is reset.

Fig. 8 shows an example of another embodiment of a method for accessing cache according to the present invention, where a method for controlling private cache is added. The example shown in Fig. 8 also uses two ways for private cache.

In Fig. 8, a processor begins access to a private cache, and step 801 performs a hit check to see if there is a hit or not. Next, step 802 branches depending on whether there is a hit or not. If there is a hit, control goes to step 814. If there is a

miss, control goes to step 803, and a hit check is performed to determined if there is a hit to the shared cache. Step 804 branches depending on whether there is a hit or not. If the shared cache is hit, control goes to step 821 and the shared cache is accessed. The operations performed for the hit access of the shared cache at step 821 is similar to the operations performed starting with step 212 from Fig. 2.

If step 804 determines that the shared cache is missed, control goes to step 822 and shared cache miss access operations are performed. The shared cache miss access operations at step 822 are similar to the operations performed stating with step 203 from Fig. 2. Then, control goes to step 805 in Fig. 8, and the ways are checked to see if there is a way with private cache having a valid bit V set to 0. If there is a way with the valid bit V set to 0, control goes to step 807, and the way with V set to 0 is selected. If there is no way with valid bit V set to 0, step 806 checks to see if the shared bit S is set to 1 for just one way. If only one way has S=1, then control goes to step 808, where the way with S=1 is selected. If both ways have S=1 or S=0, then control goes to step 809, and the way that was used earliest, based on the LRU, is selected. In embodiments using three or more ways, the way that was used earliest, based on the LRU, is selected if there are at least two ways with S=1. Next, step 810 checks the selected way to see if the dirty bit D is set to 1 or not. If the

dirty bit D is set to 1, control goes to step 811 and the contents of the way are written to the shared cache or the main storage. Then, at step 812, this data is read from the shared cache and stored in the selected way. At step 813, the valid bit V is set to 1, the shared bit S is set to 0, the dirty bit D is set to 0, and control proceeds to step 814. Step 814 checks to see if this access is read or write. If it is a read operation, control goes to step 815, and this data is output to the processor.

If the operation is a write operation, control proceeds to step 816, and the shared bit S is checked to see if it is set to 1 or not. If the shared bit S is set to 1, control goes to step 817, and cache invalidation requests for this data are output to the common bus for the other processors. Then, at step 818, the shared bit S is set to 0.

Next, at step 819, the write data output from the processor is written to the private cache. Then, the dirty bit D is set to 1 at step 820, and the operation is completed.

The following is a detailed description of the operations of the present invention, with the addition of private caches, compared with the comparative example.

Fig. 9 shows an example of a cache control method prepared for comparison

with the cache control method of the present invention, as shown in Fig. 8. In comparison with the present invention shown in Fig. 8, the steps for selecting a way based on the shared bit S, corresponding to steps 806, 808, are omitted. Furthermore, the shared cache access at steps 921, 922 are equivalent to the operations beginning with steps 412, 403 from the comparative example shown in Fig. 4.

With these differences, the present invention with private caches performs the operations described below and provides the desired advantages.

Fig. 10 and Fig. 11 are drawings for the purpose of describing the operations performed in the cache control method according to the present invention shown in Fig. 8 and the operations performed in the cache control method of the comparative example shown in Fig. 9. Fig. 10 is an example of how the present invention operates. Private caches 7a, 7b are the private caches for the processors a, b, respectively. The tag and data blocks 71a, 71b are the two blocks in the private cache 7a. The figure also shows the shared cache 4 and the main storage 2. The processor a, b access the shared area addresses 0100 - 0107, and then the processor a accesses the private area 2100 - 2107, the shared area 0100 - 0107, the private area 3100 - 3107, 2100 - 2107, and 3100 - 3107, in the sequence shown in the figure. To simplify the description, these addresses correspond to a single set in the private cache 7a. The

figure shows, for each point in time, the addresses of the main storage 2 cached in the set. In the example shown in Fig. 10, the total size of the data transferred from the main storage 2 in the sample operations shown in the figure is 24 bytes. Fig. 11 shows the operations performed by the comparative example from Fig. 9 for the same data accesses as shown in Fig. 10. In Fig. 11, the total data size transferred from the main storage 2 is 32 bytes, which is 4/3 the size from Fig. 10.

With the present invention, the cache in a multiprocessor machine can be controlled so that the data transferred between the cache and main storage is reduced. In a system where multiprocessors and cache are integrated on-chip, the data communication between the chip and external components can be reduced.

Claims

1. A multiprocessor machine comprising: a plurality of processors; a first cache shared by said plurality of processors; and a first controller providing control so that data accessed by at least two processors out of said plurality of processors is given higher priority in being saved to said first cache compared to data accessed by only one of said plurality of processors.

2. A multiprocessor machine as described in claim 1 further comprising: second caches associated with each of said plurality of processors; and a second controller providing control so that, when data stored in a second cache is accessed by a processor other than a processor associated with said second cache, said data is not stored in said second cache with a higher priority compared to data accessed only by said processor associated with said second cache.

3. A multiprocessor machine as described in claim 1 wherein said plurality of processors and said first cache and said first controller are integrated on a single LSI chip.

4. A multiprocessor machine as described in claim 2 wherein said plurality of processors and said first cache and said controller and said second cache and said second controller are integrated on a single LSI chip.

5. A multiprocessor machine as described in claim 1 wherein said first controller includes first selecting means which, if storing new data to said first cache and there is an area in said first cache containing data not accessed by at least two processors of said plurality of processors, selects said area in said first cache over an area containing data accessed by at least two processors of said plurality of processors.

6. A multiprocessor machine as described in claim 2 wherein said second controller includes second selecting means which, if storing new data to said second cache and there is an area in said second cache containing data accessed by a processor other than a processor associated with said second cache, selects said area in said second cache over an area containing data accessed by only said processor associated with said second cache.

7. A method for controlling cache comprising:

a first step evaluating whether data stored in a cache shared by a plurality of processors is accessed by at least two processors from said plurality of processors;

a second step selecting an area determined by said first step to not be accessed by at least two processors when storing new data to said cache;

a third step selecting an area in said cache if no area can be selected in said second step; and

a fourth step storing said new data in said cache area selected by either said second step or said third step.

8. A method for controlling cache as described in claim 7 wherein said third step selects an area containing data with the lowest number of accessing processors.

9. A method for controlling cache comprising:

a first step evaluating whether data stored in a cache associated with one of a plurality of processors was accessed by a processor other than a processor associated with said cache;

a second step selecting an area containing data determined in said first step

to have been accessed by another processor when new data is stored in said cache;

a third step selecting an area of said cache if no area can be selected in said second step; and

a fourth step storing new data in an area selected by either said second step or said third step.

10. A processor system comprising: a plurality of processors; a cache memory accessible by at least two processors of said plurality of processors; a first bus connecting said plurality of processors and said cache memory; a main storage memory exchanging data with said cache memory; a second bus connecting said cache memory and said main storage memory; a sharing evaluation module evaluating whether data stored in said cache memory is accessed by at least two processors and adding attributes to said data; and a replacement controller selecting data in said cache memory determined to not be accessed by at least two processors based on said attributes over data determined to be accessed by at least two processors, and replacing data in said main storage memory with said selected data.

11. A processor system as described in claim 10 wherein said cache memory includes

a plurality of sets, said sets containing information indicating whether data in said sets is use by a plurality of processors.

12. A processor system as described in claim 11 wherein said sets include information indicating validity of data and information indicating whether data was rewritten.

13. A processor system as described in claim 12 wherein said sets include an identifier of a processor that last accessed said data.

14. A processor system as described in claim 13 wherein said replacement controller selects an area of said cache memory to replace data in said main storage memory based on said information indicating whether data in said set is use by a plurality of processors, said information indicating validity of data, and said information indicating whether data was rewritten.

Abstract

In multiprocessor machines and chip multiprocessor systems in particular, the object of the present invention is to reduce data communication between the LSI chip and external components and to avoid restrictions in communication volume resulting from the LSI pin count. Sets in tag and data blocks of a shared cache include a shared bit S. When data is replaced for a cache miss, the contents of the shared bit S are checked and the side with the shared bit S set to 0 in the tag and data block is selected for data replacement. This allows data shared by a plurality of processors to be left in the shared cache, and the data transfer between the shared cache and the main memory can be reduced.

Fig. 1

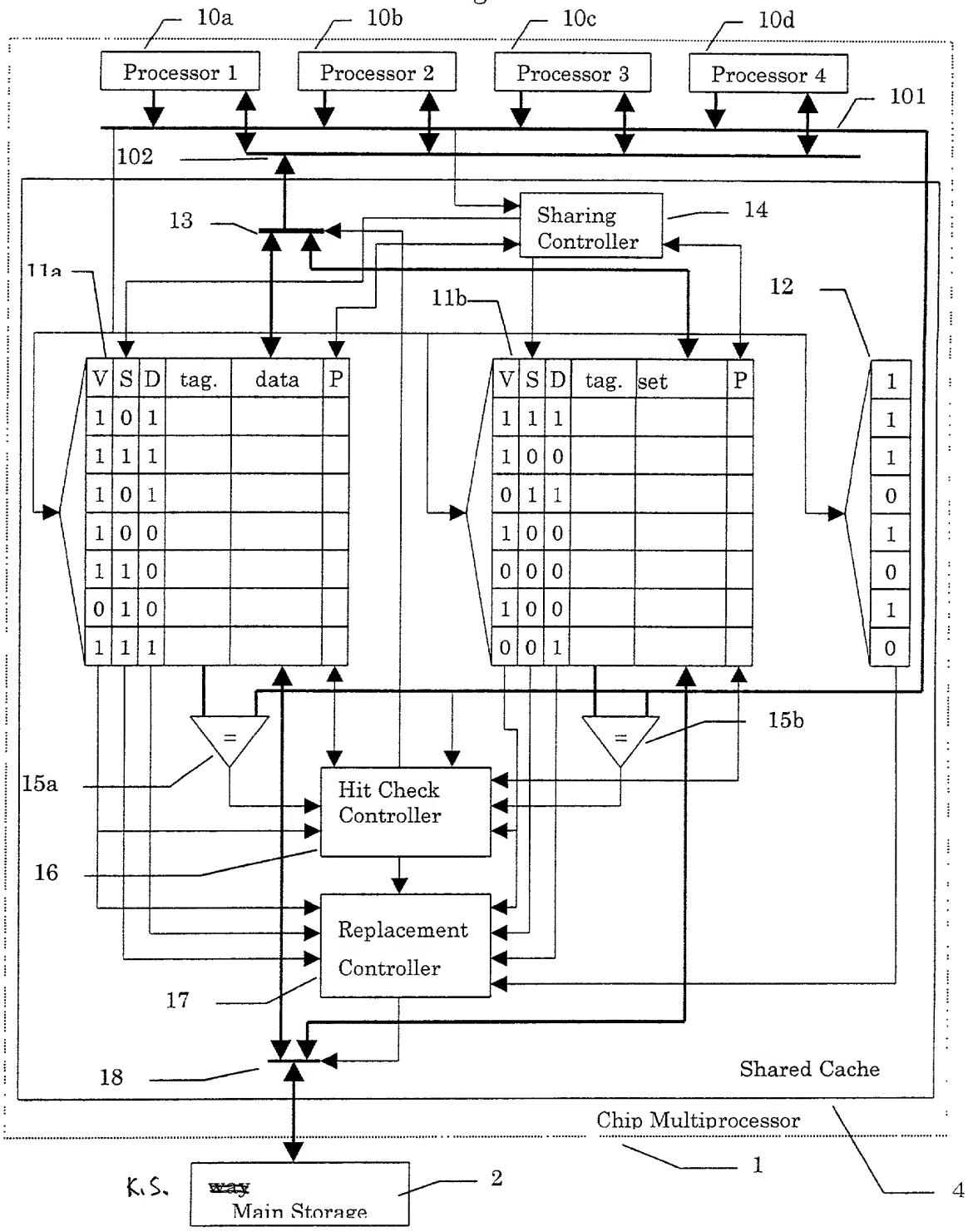

10a
10b
10c
10d
Processor 1
Processor 2
Processor 3
Processor 4
101
102
13
Sharing Controller
14
11a
11b
12
V S D tag. data P
V S D tag. set P
1
1
1
0
1
0
1
=
=
15b
15a
Hit Check Controller
16
Replacement Controller
17
18
Shared Cache
Chip Multiprocessor
1
4
K.S.
way
Main Storage
2

Fig. 2

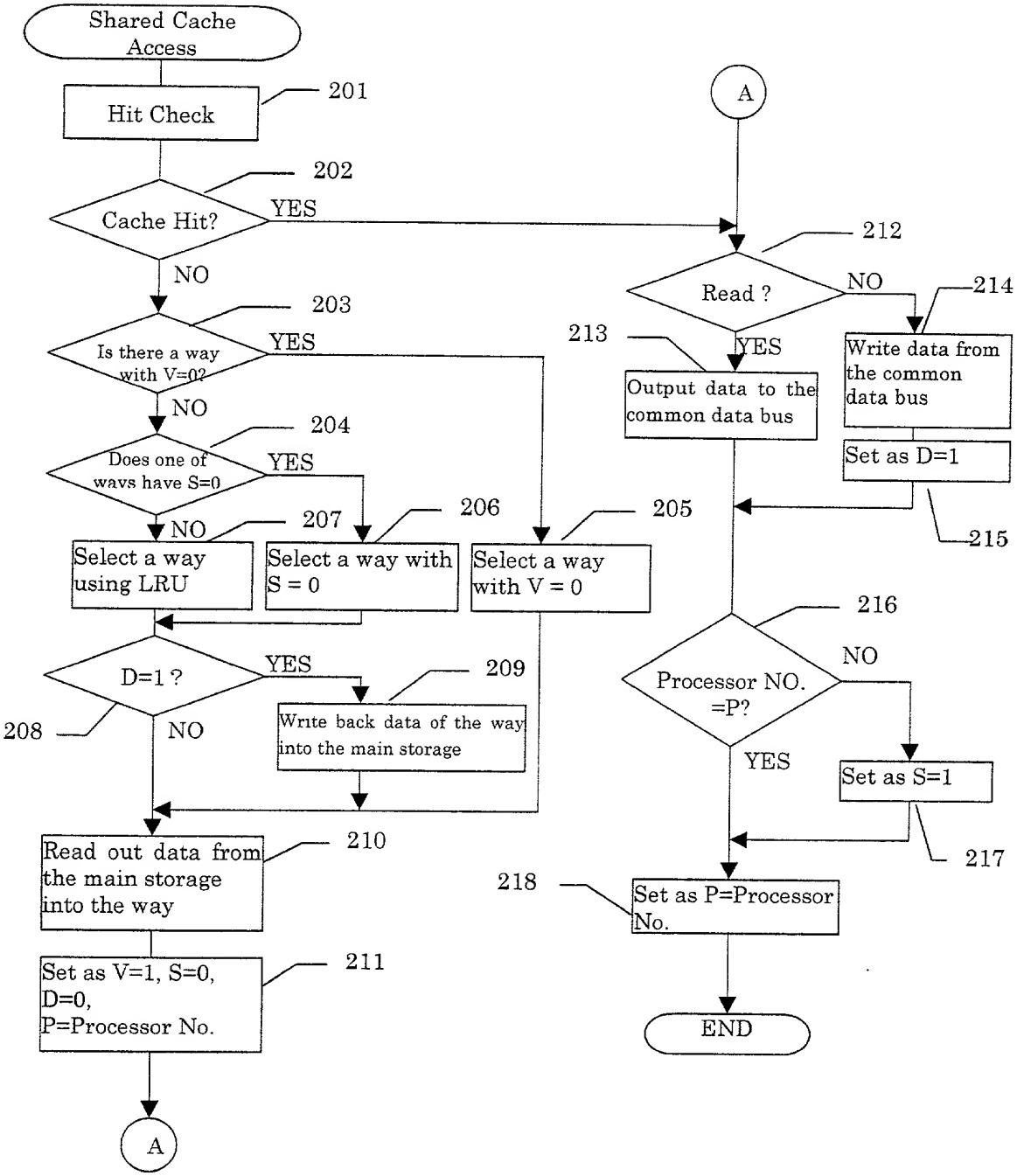

Shared Cache Access
Hit Check
201
Cache Hit?
202
YES
NO
Is there a way with V=0?
203
YES
NO
Does one of ways have S=0
204
YES
NO
Select a way using LRU
207
Select a way with S = 0
206
Select a way with V = 0
205
D=1 ?
YES
NO
208
Write back data of the way into the main storage
209
Read out data from the main storage into the way
210
Set as V=1, S=0, D=0, P=Processor No.
211
A
A
Read ?
212
NO
YES
213
Output data to the common data bus
Write data from the common data bus
214
Set as D=1
215
Processor NO. =P?
216
NO
YES
Set as S=1
217
Set as P=Processor No.
218
END

Fig. 3

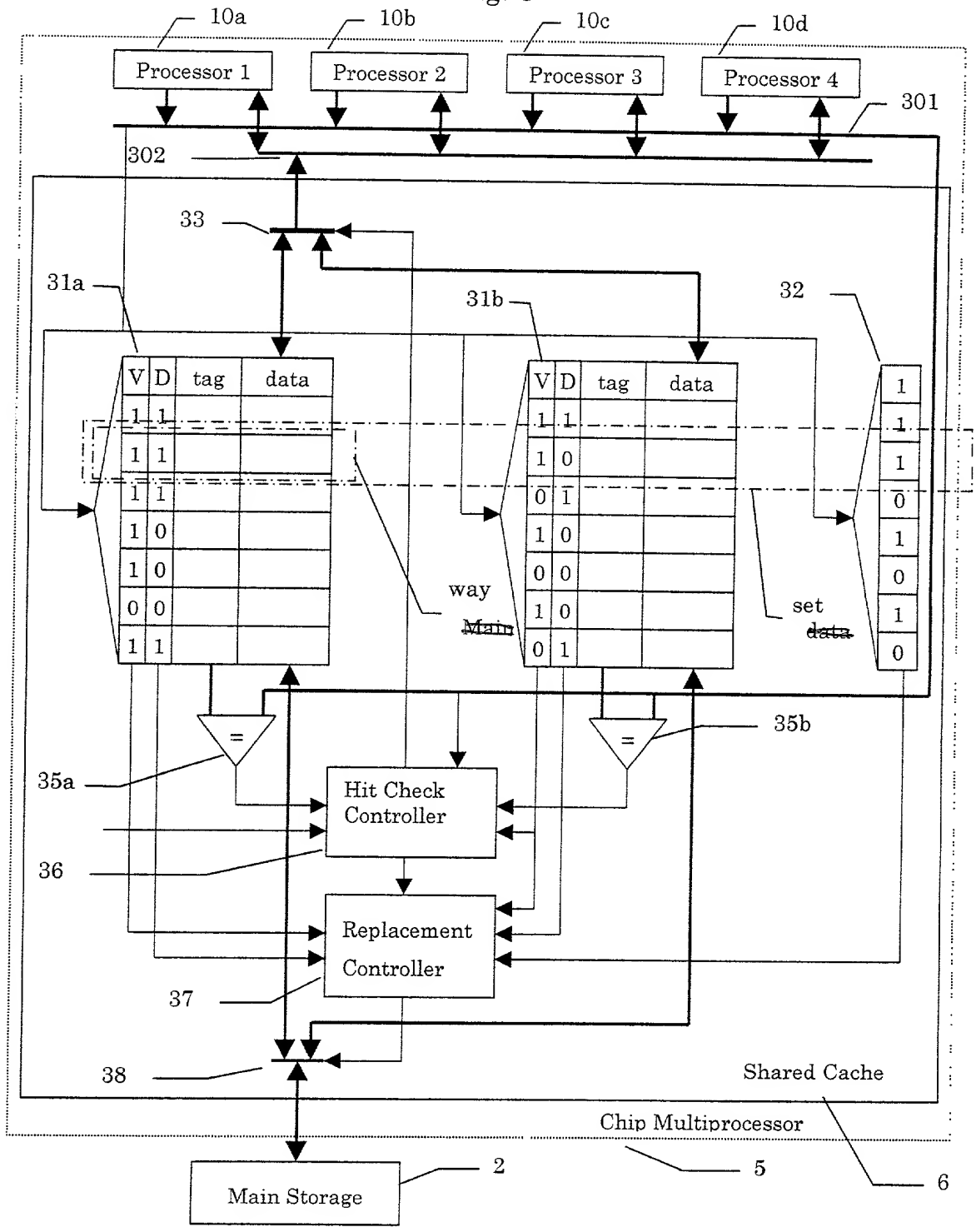

10a
10b
10c
10d
Processor 1
Processor 2
Processor 3
Processor 4
301
302
33
31a
31b
32
V
D
tag
data
way
Main
set
data
35a
35b
=
=
Hit Check Controller
36
Replacement Controller
37
38
Shared Cache
Chip Multiprocessor
Main Storage
2
5
6

Fig. 4

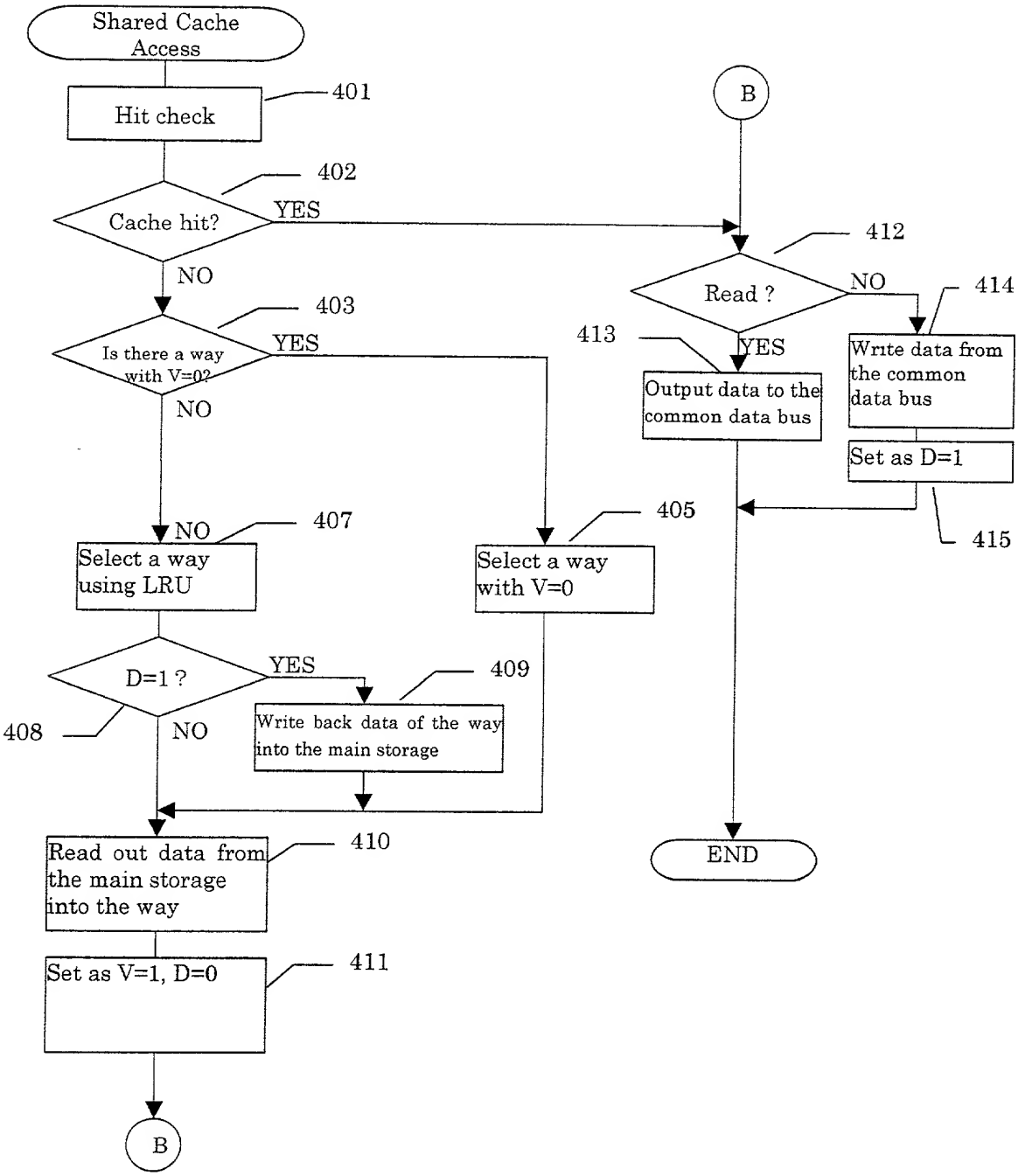

Shared Cache Access
Hit check
401
Cache hit?
402
YES
NO
Is there a way with V=0?
403
YES
NO
Select a way using LRU
407
NO
D=1 ?
YES
409
408
NO
Write back data of the way into the main storage
Read out data from the main storage into the way
410
Set as V=1, D=0
411
B
B
Read ?
412
NO
414
YES
413
Output data to the common data bus
Write data from the common data bus
Set as D=1
415
Select a way with V=0
405
END

Fig. 5

An example of behavior of said invention (Fig.1)
Time tick in parentheses

(1) Processor a: read address 0100~0107;
8 bytes data transfer from main storage 2 to tag and data block 11a.

(2) Processor b : read address 0100~0107;
read data from tag and data block 11a,
set shared bit S.

(3) Processor c : read address 2100~2107;
8 bytes data transfer from main storage 2 to tag and data block 11b.

(4) Processor d : read address 3100~3107;
8 bytes data transfer from main storage 2 to tag and data block 11b, again.

(5) Processor a : read address 0100~0107;
read data from tag and data block 11a

(6) Processor c : read address 4100~4107;
8 bytes data transfer from main storage 2 to tag and data block 11b, again.

(7) Processor d : read address 5100~5107;
8 bytes data transfer from main storage 2 to tag and data block 11b, again.

(8) Processor b : read address 0100~0107;
read data from tag and data block 11a

40 bytes data transfer in all

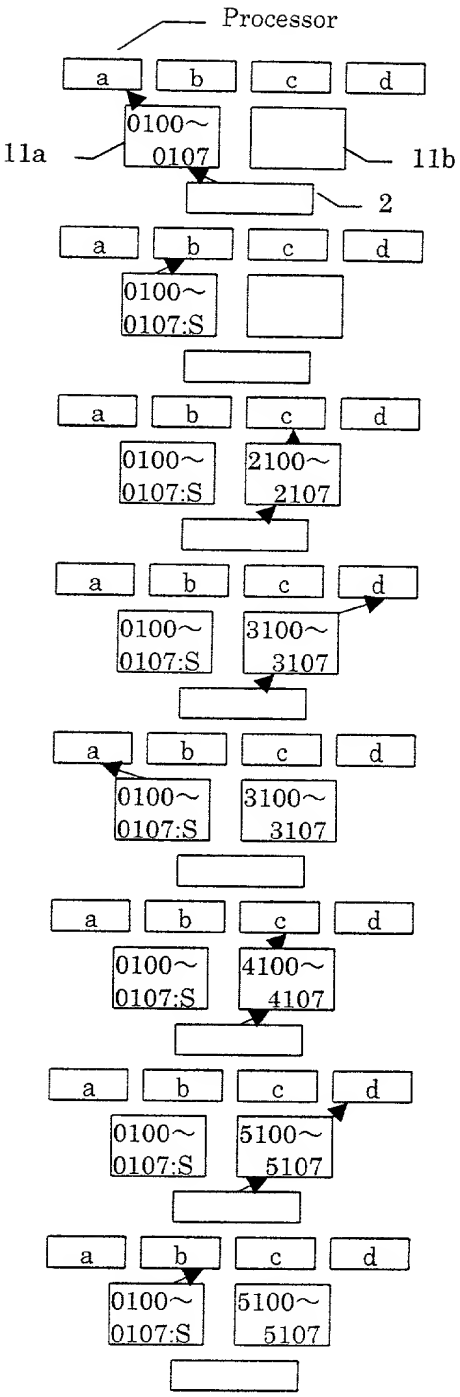


Fig. 6

An example of behavior on Fig.3
Time tick in parentheses

(1) Processor a: read address 0100~0107;
8 bytes data transfer from main storage 2 to tag and data block 31a

(2) Processor b : read address 0100~0107;
read data from tag and data block 31a

(3) Processor c : read address 2100~2107;
8 bytes data transfer from main storage 2 to tag and data block 31b

(4) Processor d : read address 3100~3107;
8 bytes data transfer from main storage 2 to tag and data block 31a

(5) Processor a : read address 0100~0107;
8 bytes data transfer from main storage 2 to tag and data block 31b

(6) Processor c : read address 4100~4107;
8 bytes data transfer from main storage 2 to tag and data block 31a

(7) Processor d : read address 5100~5107;
8 bytes data transfer from main storage 2 to tag and data block 31b

(8) Processor b : read address 0100~0107;
8 bytes data transfer from main storage 2 to tag and data block 31a

56 bytes data transfer in all

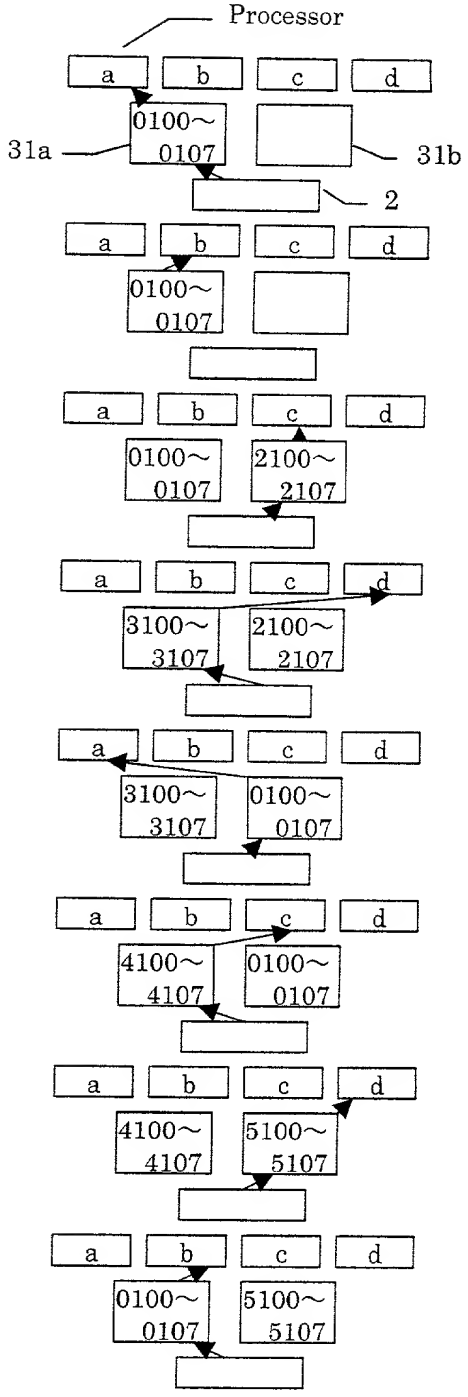


Fig. 7

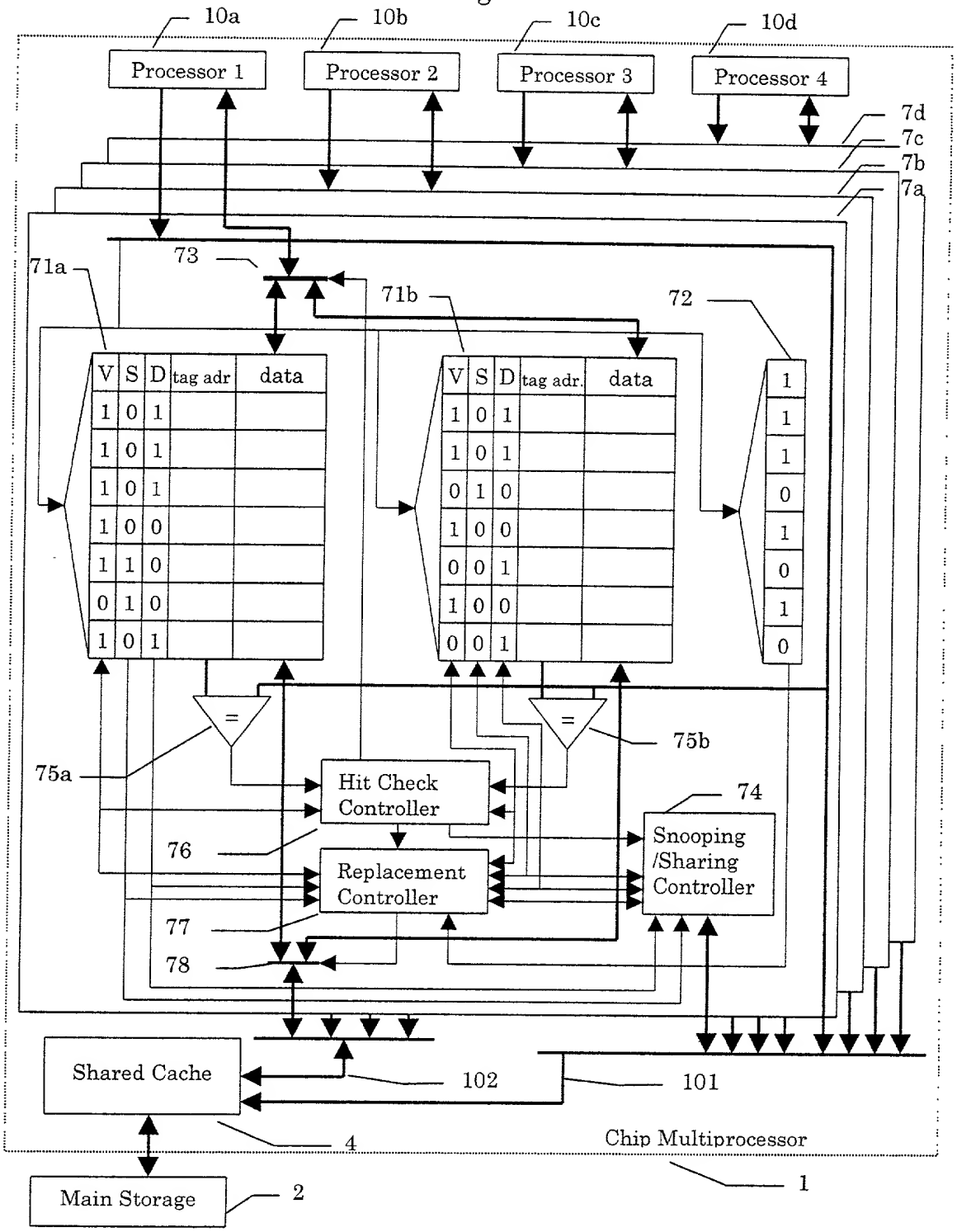

10a
10b
10c
10d
Processor 1
Processor 2
Processor 3
Processor 4
7d
7c
7b
7a
73
71a
71b
72
V S D tag adr data
1 0 1
1 0 1
1 0 1
1 0 0
1 1 0
0 1 0
1 0 1
V S D tag adr. data
1 0 1
1 0 1
0 1 0
1 0 0
0 0 1
1 0 0
0 0 1
1
1
1
0
1
0
1
0
=
=
75a
75b
Hit Check Controller
74
Snooping /Sharing Controller
76
Replacement Controller
77
78
Shared Cache
102
101
4
Chip Multiprocessor
1
Main Storage
2

Fig. 8

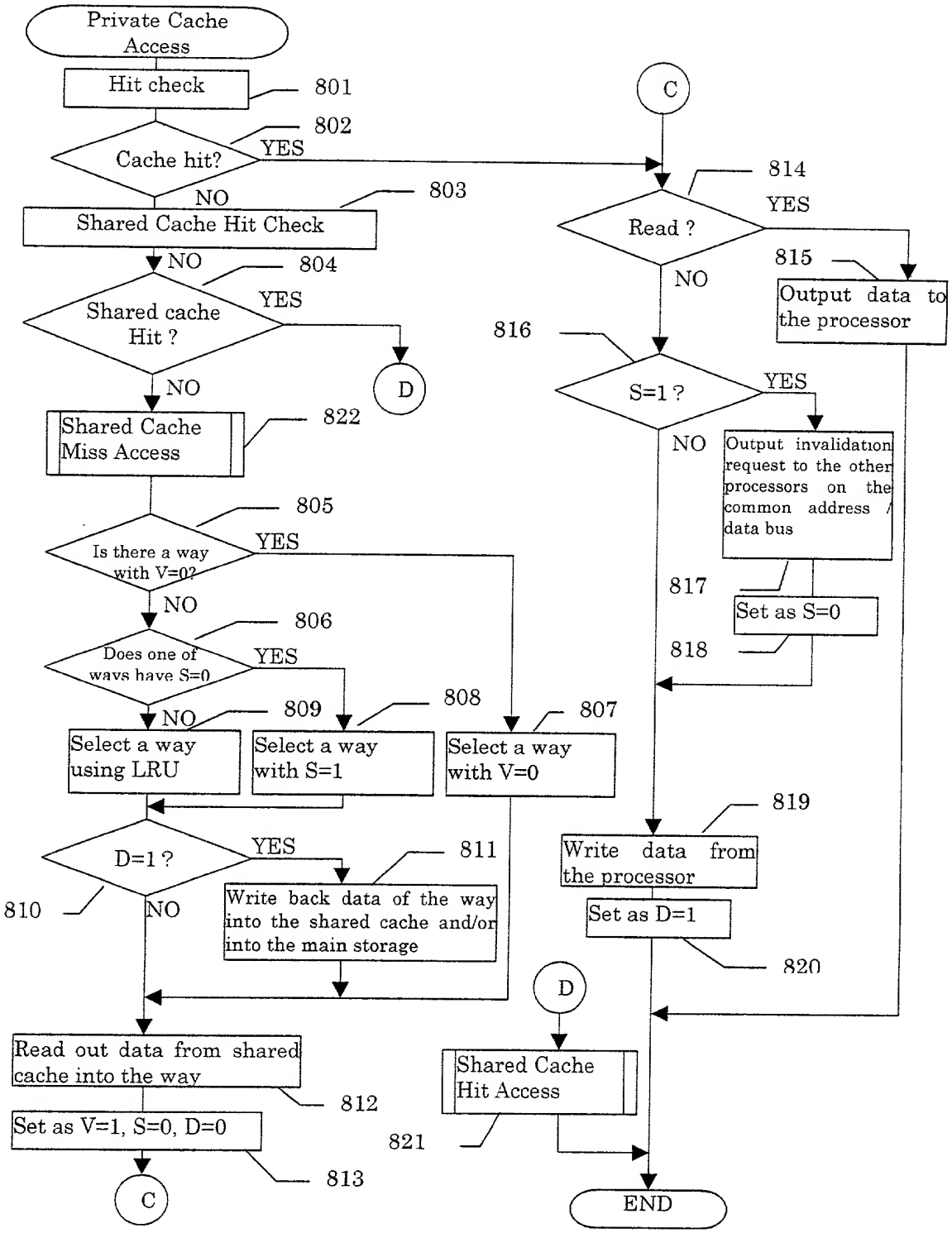

Private Cache Access
Hit check
801
Cache hit?
802
YES
NO
803
Shared Cache Hit Check
NO
804
Shared cache Hit ?
YES
NO
D
822
Shared Cache Miss Access
805
Is there a way with V=0?
YES
NO
806
Does one of ways have S=0
YES
NO
809
808
807
Select a way using LRU
Select a way with S=1
Select a way with V=0
D=1 ?
YES
811
NO
810
Write back data of the way into the shared cache and/or into the main storage
Read out data from shared cache into the way
812
Set as V=1, S=0, D=0
813
C
C
814
Read ?
YES
NO
815
Output data to the processor
816
S=1 ?
YES
NO
Output invalidation request to the other processors on the common address / data bus
817
Set as S=0
818
819
Write data from the processor
Set as D=1
820
D
Shared Cache Hit Access
821
END

Fig. 9

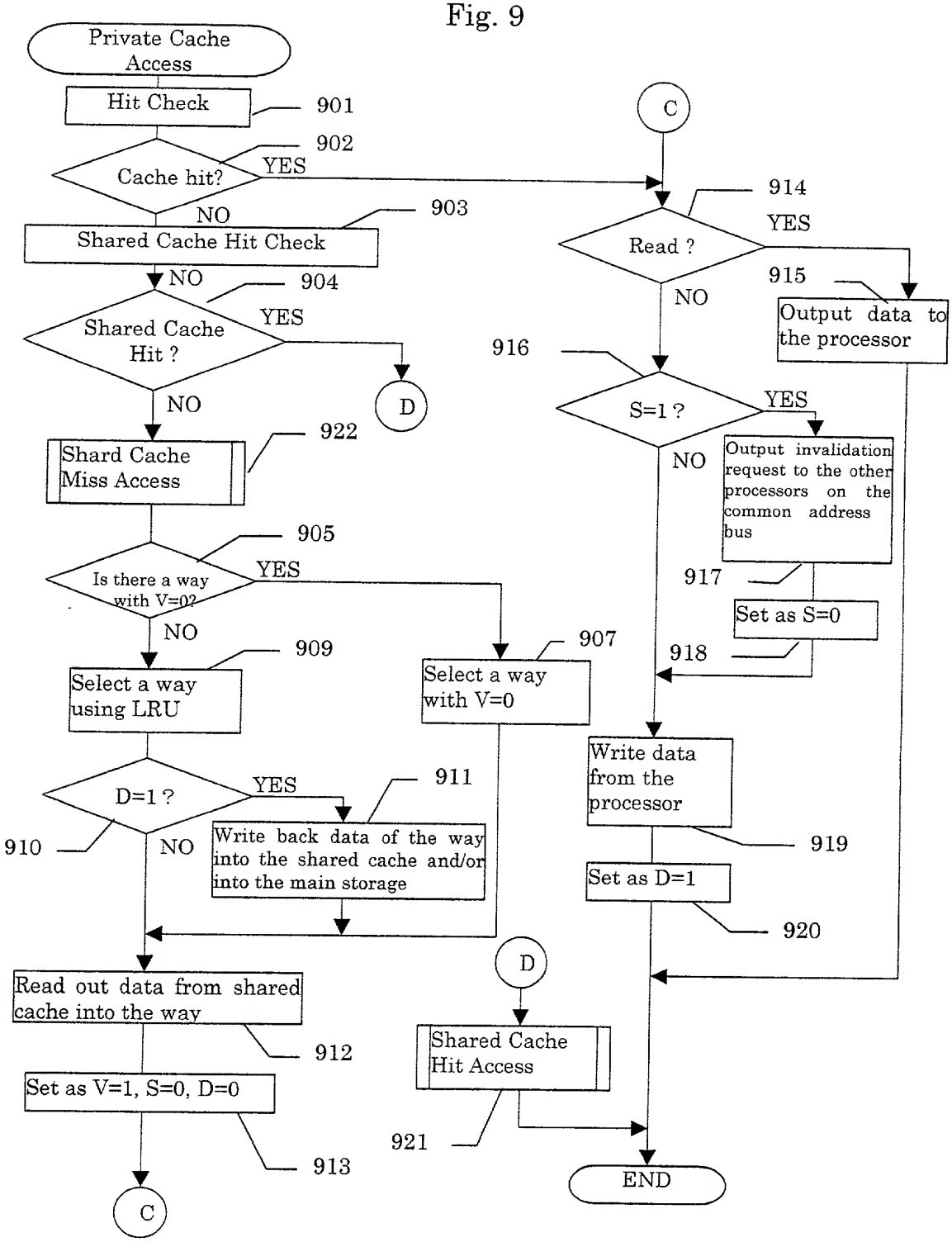

Private Cache Access
Hit Check
901
Cache hit?
902
YES
NO
Shared Cache Hit Check
903
NO
Shared Cache Hit ?
904
YES
NO
D
Shard Cache Miss Access
922
Is there a way with V=0?
905
YES
NO
Select a way using LRU
909
D=1 ?
YES
911
910
NO
Write back data of the way into the shared cache and/or into the main storage
Read out data from shared cache into the way
912
Set as V=1, S=0, D=0
913
C
C
914
Read ?
YES
NO
915
Output data to the processor
916
S=1 ?
YES
NO
Output invalidation request to the other processors on the common address bus
917
Set as S=0
918
907
Select a way with V=0
Write data from the processor
919
Set as D=1
920
D
Shared Cache Hit Access
921
END

Fig. 10

An example of behavior of private cache of said invention (Fig.8)
Time tick in parentheses

(1) Processor a : read address 0100~0107;
8 bytes data transfer from main storage 2 to shared cache 4(tag and data block 11a) and to private cache 7a(tag and data block 71a)

(2) Processor b : read address 0100~0107;
8 bytes data transfer from shared cache 4 to private cache 7b, set shared bit S in private cache 7a(tag and data block 71a)

(3) Processor a : read address 2100~2107;
8 bytes data transfer from main storage 2 to shared cache 4(tag and data block 11b) and to private cache 7a(tag and data block 71b)

(4) Processor a : read address 0100~0107;
read data from private cache 7a(tag and data block 71a)

(5) Processor a : read address 3100~3107;
8 bytes data transfer from main storage 2 to shared cache 4(tag and data block 11b) and to private cache 7a(tag and data block 71a)

(6) Processor a : read address 2100~2107;
read data from private cache 7a(tag and data block 71b)

(7) Processor a : read address 3100~3107;
read data from private cache 7a(tag and data block 71a)

24 bytes data transfer in all

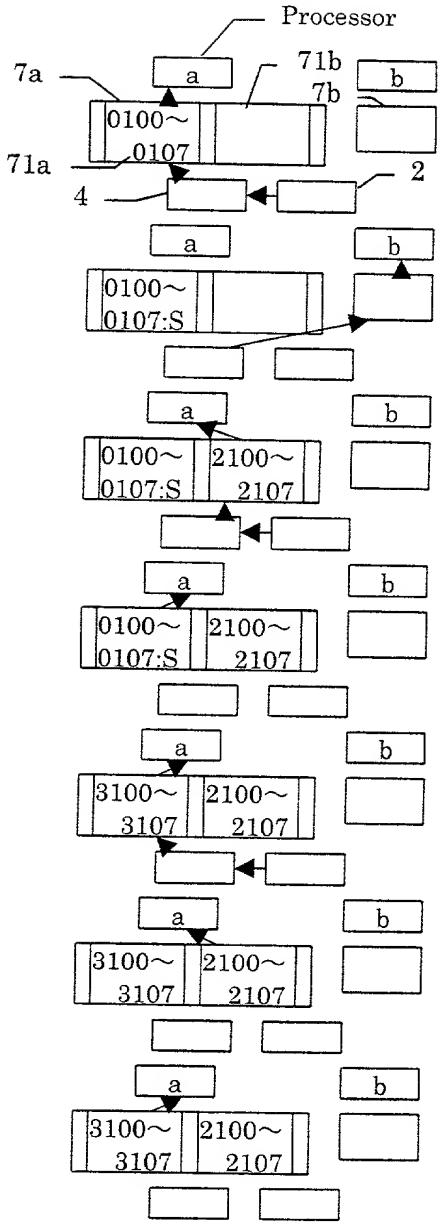


Fig. 11

An example of behavior of private cache on Fig.8

Time tick in parentheses

(1) Processor a : read address 0100～0107;
8 bytes data transfer from main storage 2 to shared cache 4(tag and data block 11a) and to private cache 7a(tag and data block 71a)

(2) Processor b : read address 0100～0107;
8 bytes data transfer from shared cache 4 to private cache 7b

(3) Processor a : read address 2100～2107;
8 bytes data transfer from main storage 2 to shared cache 4(tag and data block 11b) and to private cache 7a(tag and data block 71b)

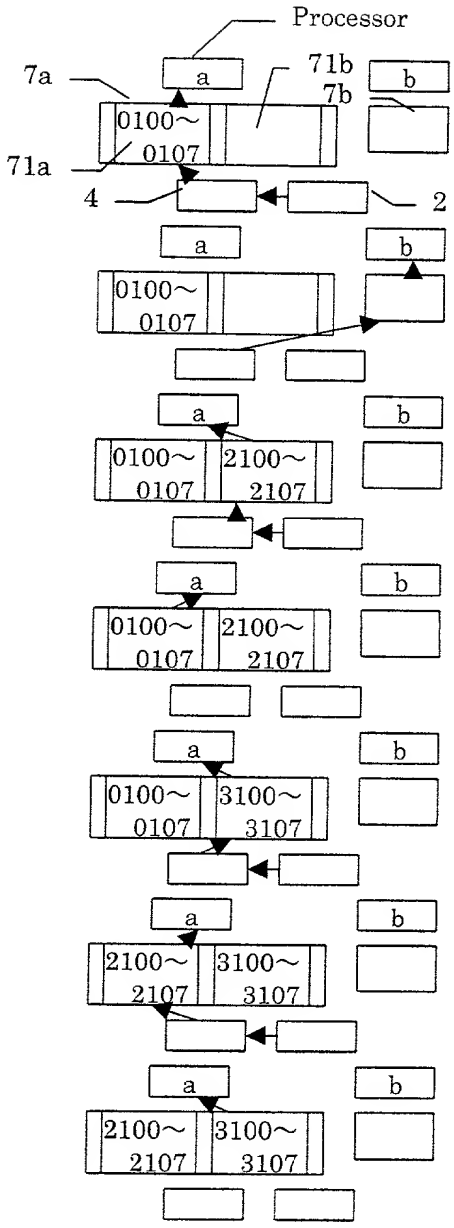
(4) Processor a : read address 0100～0107;
read data from private cache 7a(tag and data block 71a)

(5) Processor a : read address 3100～3107;
8 bytes data transfer from main storage 2 to shared cache 4(tag and data block 11b) and to private cache 7a(tag and data block 71b)

(6) Processor a : read address 2100～2107;
8 bytes data transfer from main storage 2 to shared cache 4(tag and data block 11b) and to private cache 7a(tag and data block 71a)

(7) Processor a : read address 3100～3107;
read data from private cache 7a(tag and data block 71b)

32 bytes data transfer in all



COMBINED DECLARATION AND POWER OF ATTORNEY

As a below named inventor, I hereby declare that:

My residence, post office address and citizenship are as stated next to my name. I believe I am the original, first and sole inventor (if only one name is listed below), or an original, first and joint inventor (if plural names are listed below), of the subject matter claimed and for which a patent is sought on the invention entitled:

MULTIPROCESSOR MACHINE AND CACHE CONTROL METHOD

the specification of which: (check one) [X] is attached hereto.

[ ] was filed on _____

as Application Serial No. _____

and was amended on _____
(if applicable)

I hereby state that I have reviewed and understand the contents of the above identified specification, including the claims, as amended by any amendment referred to above.

I acknowledge the duty to disclose information material to examination of this application according to Title 37, Code of Federal Regulations, Section 1.56.

I hereby claim foreign priority benefits under Title 35, United States Code, Section 119(a)-(d) or Section 365(b)of any foreign application (s) for patent or inventor's certificate, or Section 365(a) of any PCT International application which designated at least one country other than the United States, listed below and have also identified below, by checking the box, any foreign application for patent or inventor's certificate, or PCT International application having a filing date before that of the application on which priority is claimed:

Prior Foreign Application(s)

			Priority Claimed	
_____	_____	_____	[ ]	[ ]
(Number)	(Country)	(Day/Month/Year Filed)	Yes	No
_____	_____	_____	[ ]	[ ]
(Number)	(Country)	(Day/Month/Year Filed)	Yes	No
_____	_____	_____	[ ]	[ ]
(Number)	(Country)	(Day/Month/Year Filed)	Yes	No

I hereby claim the benefit under Title 35, United States Code, Section 119(e) of any United States provisional application(s) listed below.

_____	_____	_____
(Application Number)	(Filing Date)	(Status -- patented, pending, abandoned)
_____	_____	_____
(Application Number)	(Filing Date)	(Status -- patented, pending, abandoned)

I hereby claim the benefit under Title 35, United States Code, Section 120, of any United States application(s) or Section 365(c) of any PCT International application designating the United States, listed below and, insofar as the subject matter of each of the claims of this application is not disclosed in the prior United States or PCT International application in manner provided by the first paragraph of Title 35, United States Code, Section 112, I acknowledge the duty to disclose information which is material to patentability as defined in Title 37, Code of Federal Regulations, Section 1.56 which became available between the filing date of the prior application and the national or PCT International filing date of this application:

(Application Serial No.)	(Filing Date)	(Status) (patented, pending, abandoned)
(Application Serial No.)	(Filing Date)	(Status) (patented, pending, abandoned)
(Application Serial No.)	(Filing Date)	(Status) (patented, pending, abandoned)

I hereby appoint the following attorneys/agents to prosecute this application and transact all business in the Patent and Trademark Office connected therewith and with any divisional, continuation, continuation-in-part, reissue or re-examination application with full power of appointment and substitution of associate attorneys and agents, and to receive all patents which may issue thereon: John R. Mattingly, Reg. No. 30,293; Daniel J. Stanger, Reg. No. 32,846; Shrinath Malur, Reg. No. 34,663; Gene W. Stockman, Reg. No. 21,021; Jeffrey M. Ketchum, Reg. No. 31,174; Scott W. Brickner, Reg. No. 34,553. Address all correspondence to:

MATTINGLY, STANGER & MALUR, P.C.
104 East Hume Avenue
Alexandria, Virginia 22301
Tel. 703-684-1120

I declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further, that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Title 18, United States Code, Section 1001, and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

Date 9/19/00 Inventor Kentaro SHIMADA Kentaro Shimada
(Typed Name and Signature)
Residence 3-33-13-502, Izumi, Suginami-ku, Tokyo, 168-0063 JAPAN Citizenship Japan

Post Office Address 3-33-13-502, Izumi, Suginami-ku, Tokyo, 168-0063 JAPAN

Date _____ Inventor _____
(Typed Name and Signature)
Residence _____ Citizenship _____

Post Office Address _____